1 팀 임베디드 시스템 최종 보고서

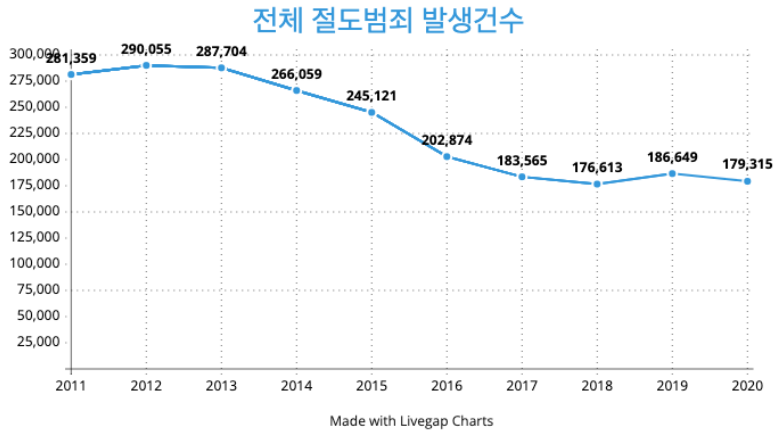
201812214 이성현

201811420 김동건

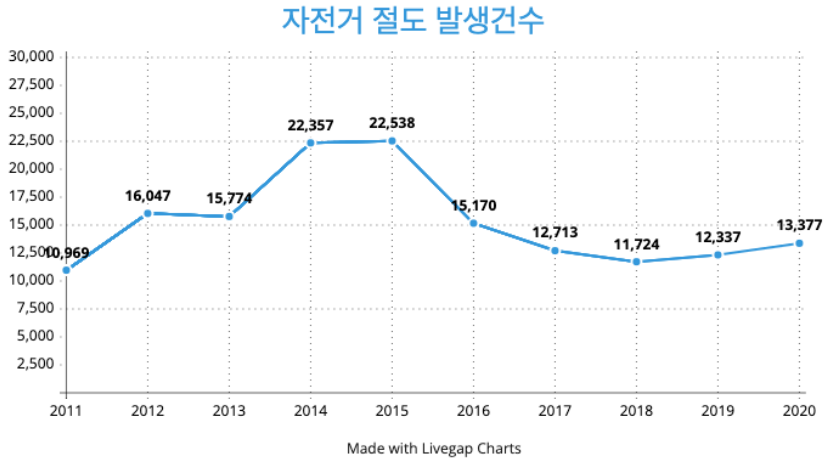
목차

1. 시스템 설계 목적

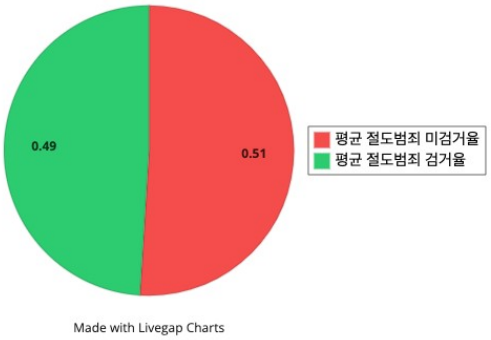
(1) 설계 동기



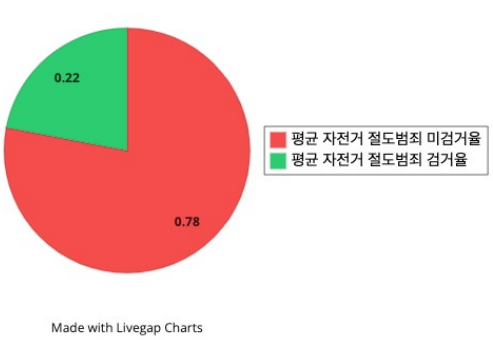
(그림 1) 전체 절도범죄 발생건수



(그림 2) 자전거 절도범죄 발생건수



(그림 3) 전체 절도범죄 검거율



(그림 4) 자전거 절도범죄 검거율

우리나라는 고도의 과학 기술의 발전과 CCTV 의 높은 보급률로 절도 범죄 발생 건수는 점점 줄어드는 추세를 보인다. 절도 범죄는 2011 년 약 28 만 건에서 2020 년 약 18 만 건으로 약 10 만 건이 감소하였다.

그러나, 자전거 절도 범죄의 경우, 2011 년 약 1 만 1 천 건에 비하여 2020 년 약 1 만 3 천 건으로 오히려 증가하는 경향을 보인다.
이는 자전거의 특징상 가격의 폭이 다양하여 상당히 고가인 물품도 존재하고, 야외에 배치되어 손쉽게 접근할 수 있다는 점과 자전거 전부를 훔치지 않더라도 주요 부품들을 분해하여 훔칠 수 있다는 점 등이 자전거 절도 발생 건수의 증가 요인으로 보인다.

또한 전체 절도 범죄에 대한 검거율에 비해 자전거 절도 범죄 검거율은 상당히 떨어지는 모습을 보여 절도 피해를 당하더라도 가해자를 잡기 힘든 상황을 잘 보여주고 있다.
따라서 우리는 자전거 절도 방지 시스템을 설계하기로 하였다.

(2) 시중 제품 분석

(그림 5) 무겁고 튼튼한 자물쇠
(자물쇠 A)

브랜드	락 브라더스	품명	미니 헬멧 자물쇠
재질	아연 합금 잠금 본체	무게	약 45g
치수	약 90cm (신축성)	컬러	빨강, 녹색, 흰색, 검정

(그림 6) 가볍고 안정성이 낮은 자물쇠
(자물쇠 B)

	무게	안정성	휴대성
자물쇠 A	무거움	높음	낮음
자물쇠 B	가벼움	낮음	높음
도난 방지 시스템	가벼움	높음	높음

(그림 7) 자물쇠 A 와 자물쇠 B 의 비교

현재 시중에 판매되는 자전거 자물쇠는 두 가지로 분류할 수 있다.

첫 번째는 무겁고 튼튼한 자물쇠로(자물쇠 A), 높은 안정성을 가져 도난 사고에 대해 좀 더 안전하지만, 무겁다는 점과 큰 크기로 인해 자전거 라이딩 시 휴대하기가 까다로워 자전거 자물쇠로 적합하지 않다고 할 수 있다.

자전거를 취미생활로 즐기는 사람들은 장기간 자전거 주차가 필요한 경우에는 보통 본인이 확인할 수 있는 실내, 본인의 집과 같은 안전한 환경에 배치해 두기 때문에 무겁고 튼튼한 자물쇠가 필요 없고, 실제 야외에 주차가 필요한 경우에는 식사와 같은 잠깐 자리를 비워야 하는 경우가 많기 때문이다.

또한, 자전거의 중요한 요소 중 하나는 무게이며, 무게를 줄임으로써 얻는 이득(가속도, 항속 유지 등이 많기 때문에 취미 생활을 즐기는 많은 사람이 최대한 경량화를 위해 노력하는 편이다.
따라서 이러한 사람들에게 자물쇠 A 는 사용하기 꺼려지는 제품이라고 할 수 있다.

두 번째는 가볍고 안정성이 낮은 자물쇠로(자물쇠 B), 가볍고 크기가 작아 높은 휴대성이 장점이지만, 두께가 얇기 때문에 자물쇠 A 에 비하여 쉽게 절단된다는 단점이 있다. 그래도 크기가 작고 무게가 가볍다는 장점이 존재하여 도난 위험에도 불구하고 이러한 제품들을 사용하는 사람들이 많다.

따라서 우리는 자물쇠 B 의 휴대성이 높고 가벼운 장점과 아두이노를 결합하여 가볍고 휴대성이 좋으면서도 높은 보안성을 유지할 수 있는 자전거 도난 방지 시스템을 구현하였다.

---

2. 시스템 구조

(1) 전체 시스템 구조

(1) - 1. 아두이노 시스템 구조

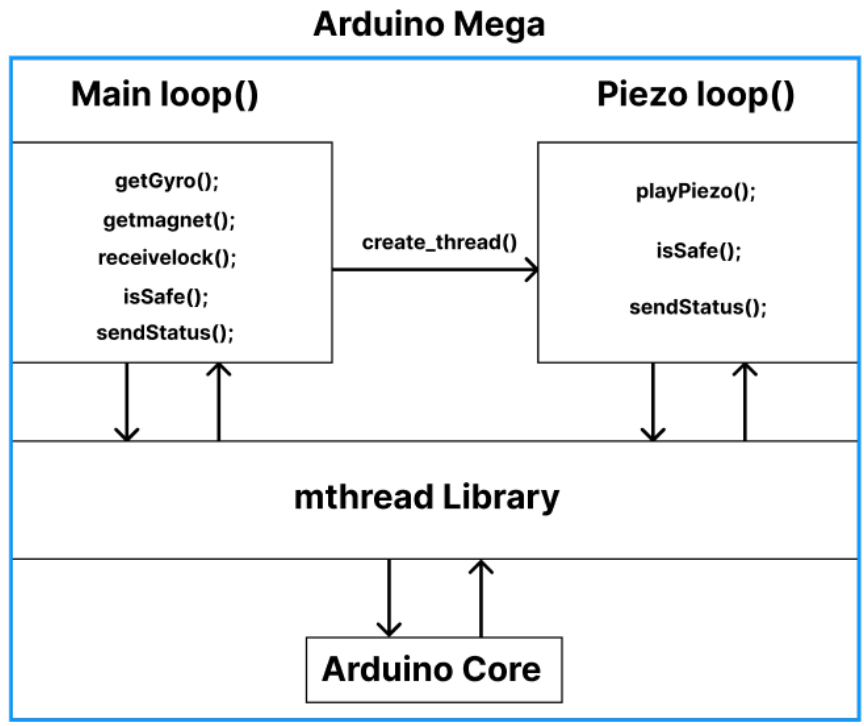


그림 8

아두이노 메가의 전체 시스템 구조는 그림 8 과 같다.
아두이노 메가에서는 도난 상황 발생시 피에조 부저를 울리는데, 이때 적절한 소리를 내기 위해 피에조 부저 함수 고유의 loop() 함수가 필요하다.

그러나 아두이노의 코어는 한개이므로, 여러개의 loop() 함수를 context switching 을 수행하는 라이브러리 mthread Library 를 이용해 피에조 부저 함수만의 loop() 함수를 생성 및 실행 한다.
Main loop() 함수 에서는 자전거의 잠금 상태를 확인하는 receivelock() 함수를 이용해 사용자의 어플리케이션으로 부터 정보를 받고 자전거의 가속도 정보는 getGyro() 함수를 이용해 MPU 6050 센서로부터 값을 읽어온다.

getmagnet() 함수를 이용해 자석 센서의 정보를 읽고, isSafe() 함수를 통해 현재 상태가 안전한지 계산한다. 만약 현재 상태의 문제가 있는 경우 sendStatus() 함수를 통해 사용자 어플리케이션으로 경보를 전송한다.

(1) - 2. 안드로이드 앱 시스템 구조

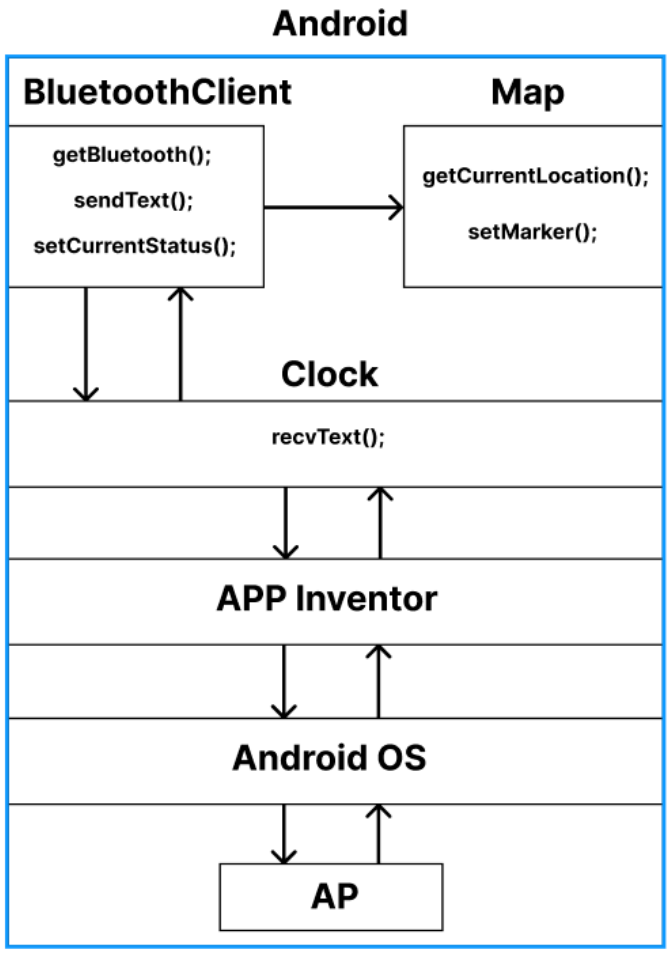


그림 9

안드로이드의 전체 시스템 구조는 그림 9 와 같다. 안드로이드는 App Inventor 에 내장된 User Interface 와 추상화된 Blocks 를 이용해 손쉽게 구성 한다. 전체를 관리하는 내장된 Thread 는 Clock Block 을 사용했다.

Clock Block 과 BluetoothClient Block 을 이용해 Arduino 와 연결해 실시간으로 값을 송/수신 했다. BluetoothClient Block 은 getBluetooth() 함수를 이용해 블루투스 연결을 수행하고, sendText() 함수를 이용해 Arduino 로 값을 전송한다.

블루투스를 통해 읽은 값을 setCurrentStatus() 함수가 처리하는데, App Inventor 에 내장된 Map 을 처리한다.

Map Block 은 getCurrentLocation() 함수를 통해 사용자 Android 휴대폰의 현재 위치 정보를 가져온다. setMarker() 함수를 이용해 현재 위치를 Map Block 에 표현한다.

(2) 시스템 상세 구조

(2) - 1. 아두이노 시스템 상세 구조

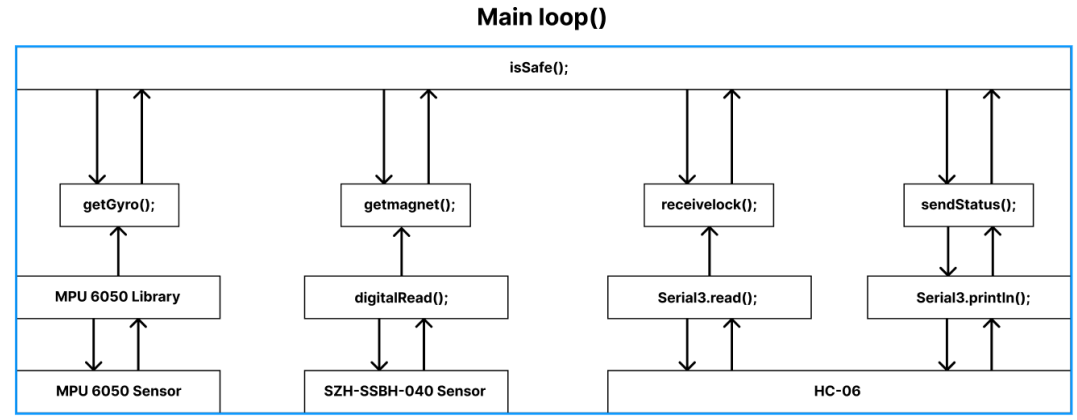


그림 10 Main loop() 구조

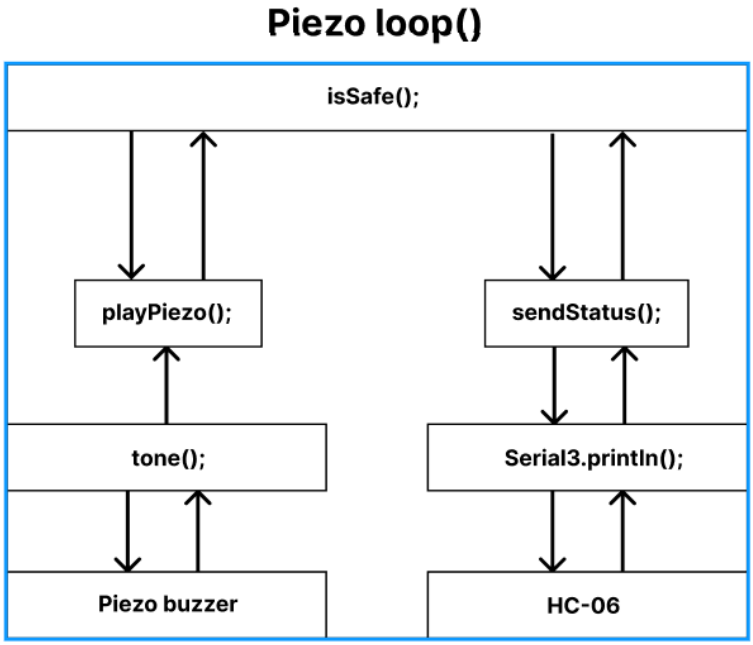


그림 11 Piezo loop() 구조

Arduino 는 Main loop(), Piezo loop() 두개의 loop() 함수로 구성 되어 있다. Main loop() 함수에서는 isSafe() 함수가 getter() 함수로부터 센서 값을 받아오고, receivelock() 함수를 통해 잠금 상태를 확인후 현재 상태에 대한 계산을 수행한다. 계산 후 값에 변화가 없는 경우 sendStatus() 함수를 통해 안전 상태를 전송한다. 만약 일정 값 이상으로 변화가 생기면 sendStatus() 함수를 통해 사용자 Android 로 현재 상태(도난 상태)를 전송한다.

또한 mthread 라이브러리의 main_thread_list->add_thread() 함수를 이용해 Piezo loop() 함수를 생성한다. Piezo loop() 함수는 playPiezo() 함수를 호출해 부저를 울리고, isSafe() 함수를 통해 현재 잠금 상태를 확인한 후 잠금이 해제되면 sendStatus() 함수를 통해 현재 상태(안전 상태)를 전송한다.

3. 시스템 구현

(1) Arduino 시스템 구현

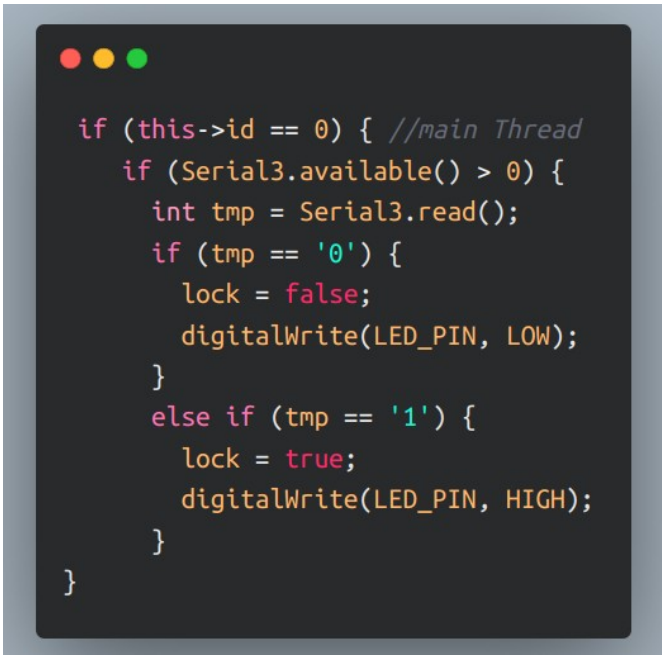
(1) - 0. mthread 라이브러리 핵심 코드 및 설명

```
#include <mthread.h>
class LoopThread : public Thread {
  public:
    LoopThread(int id);
    static int threadCount;
    static bool isPiezo;
  protected:
    bool loop();
  private:
    int id;
};

void setup() {
  main_thread_list->add_thread(new LoopThread(0));
  //Create Main Thread
}
```

```
static bool LoopThread::LoopThread::isPiezo = false;
static int LoopThread::LoopThread::threadCount = 0;

LoopThread::LoopThread(int id) {
  this->id = id;
  threadCount++;
  if (this->id == 0) { //main Thread
    Serial3.begin(9600);
    if (!mpu.begin()) {
      Serial3.println("Failed to find MPU6050");
    } else Serial3.println("MPU6050 Ready");

    mpu.setHighPassFilter(MPU6050_HIGHPASS_0_63_HZ); //센서 필터 설정
    mpu.setMotionDetectionThreshold(1); //모션감지 감도
    mpu.setMotionDetectionDuration(20); //모션감지 간격
    mpu.setMotionInterrupt(true); //모션감지 설정

    pinMode(LED_PIN, OUTPUT);
    pinMode(MAGNET, INPUT);
  }
}
```

mthread 라이브러리를 설치하고 setup() 함수에서 main_thread_list->add_thread() 함수를 이용해
thread id 가 0 인 main loop()를 생성한다.
LoopThread 의 생성자는 setup() 함수와 같은 역할을 수행하므로 블루투스 Serial3 를 초기화 하고 MPU 6050 라이브러리의 센서 필터와 감도, 간격을 설정하고 LED 와 자석 센서를 위한 pinMode 를 설정한다.

(1) - 1. Main loop() 핵심 코드 및 설명



```
if (this->id == 0) { //main Thread
   if (Serial3.available() > 0) {
     int tmp = Serial3.read();
     if (tmp == '0') {
       lock = false;
       digitalWrite(LED_PIN, LOW);
     }
     else if (tmp == '1') {
       lock = true;
       digitalWrite(LED_PIN, HIGH);
     }
}
```

Serial3.available() 함수를 이용해 안드로이드 블루투스로부터 값을 읽어 잠금/잠금 해제 상태를 설정한다.

```
if (lock == true && flag == false) { //잠금을 거는 상태
      sensors_event_t a, g, temp; //센서값 변수선언
      mpu.getEvent(&a, &g, &temp); //센서값 갱신;

      loc.x = g.gyro.x;
      loc.y = g.gyro.y;
      loc.z = g.gyro.z;
      loc.isMagnetic = digitalRead(MAGNET);
      Serial3.println(NORMAL);
      flag = true;
}
```

위의 코드로 잠금 상태 lock 이 true 가 되었다면 자이로 센서 값, 자석 센서 값을 저장하고 Android 로 정상 상태를 전송 한다. 그리고 성능 최적화를 위한 flag 를 활성화 한다.

lock	flag	상태	작업
T	T	잠금이 걸린 상태	모든 센서의 값을 읽고 계산하고 이상이 생기면 전송한다.
T	F	잠금을 거는 상태	모든 센서의 값을 읽고 현재 상태를 저장한후 정상 상태를 전송한다.
F	X	잠금이 해제된 상태	블루투스 값만 읽는다.

```
if (lock == true && flag == true) { //잠금이 걸려있는 상태
      sensors_event_t a, g, temp; //센서값 변수선언
      mpu.getEvent(&a, &g, &temp); //센서값 갱신;

      float x = g.gyro.x;
      float y = g.gyro.y;
      float z = g.gyro.z;

      if (abs(loc.x - x) >= diff) { //x값 이상
        isPiezo = true;
      } else if (abs(loc.y - y) >= diff) { //y값 이상
        isPiezo = true;
      } else if (abs(loc.z - z) >= diff) { //z값 이상
        isPiezo = true;
      }
      if (digitalRead(MAGNET) != loc.isMagnetic) {//자석 값 이상
        isPiezo = true;
      }
      if ((LoopThread::LoopThread::threadCount < 2) && (isPiezo)) {
        //Send to Android URGENCY
        Serial3.println(URGENCY);
        main_thread_list->add_thread(new LoopThread(1)); // create piezo Thread
      }
}
```

잠금이 걸린 상태라면 현재 센서값을 모두 읽고 정해진 기준값과 비교한다. 차이가 일정 이상이거나 변화가 생긴다면 thread id 가 1 인 피에조 thread 를 생성하고 Android 로 비상 상태 코드를 전송한다.

```
if (lock == false) { //lock이 해제되면 flag도 false로 만든다.
      flag = false;
}
```

lock 이 해제되면 flag 도 false 로 만든다.

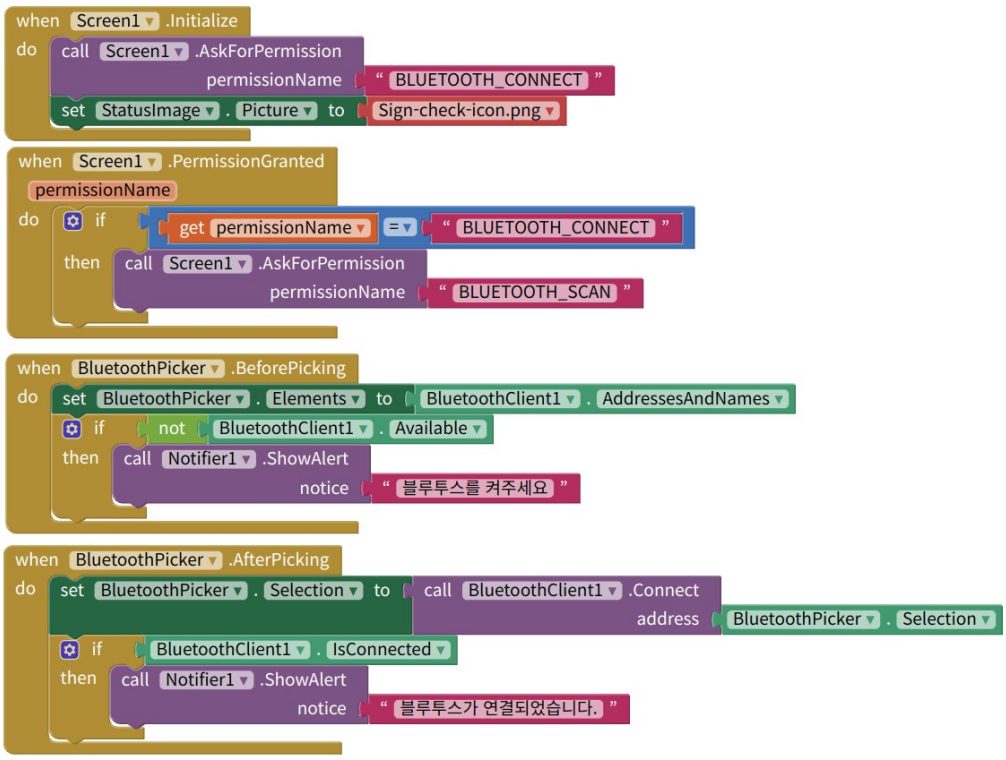
(1) - 2. Piezo loop() 핵심 코드 및 설명

```
if (this->id == 1) { //piezo Thread
    if (kill_flag || lock == false) { //Check the kill_flag OR lock
      //terminate Piezo Thread
      threadCount--;
      noTone(piezo); //Piezo Turn Off
      isPiezo = false;
      //Send to Android NORMAL state
      Serial3.println(NORMAL);
      return false;
    }
 //피에조 부저 소리
    int toneVal;
    float sinVal;
    for (int x = 0; x < 360; x++) {
      sinVal = (sin(x * (3.1412 / 180)));
      toneVal = 2000 + (int(sinVal * 1000));
      tone(12, toneVal);
    }
    noTone(12);
    return true;
}
```

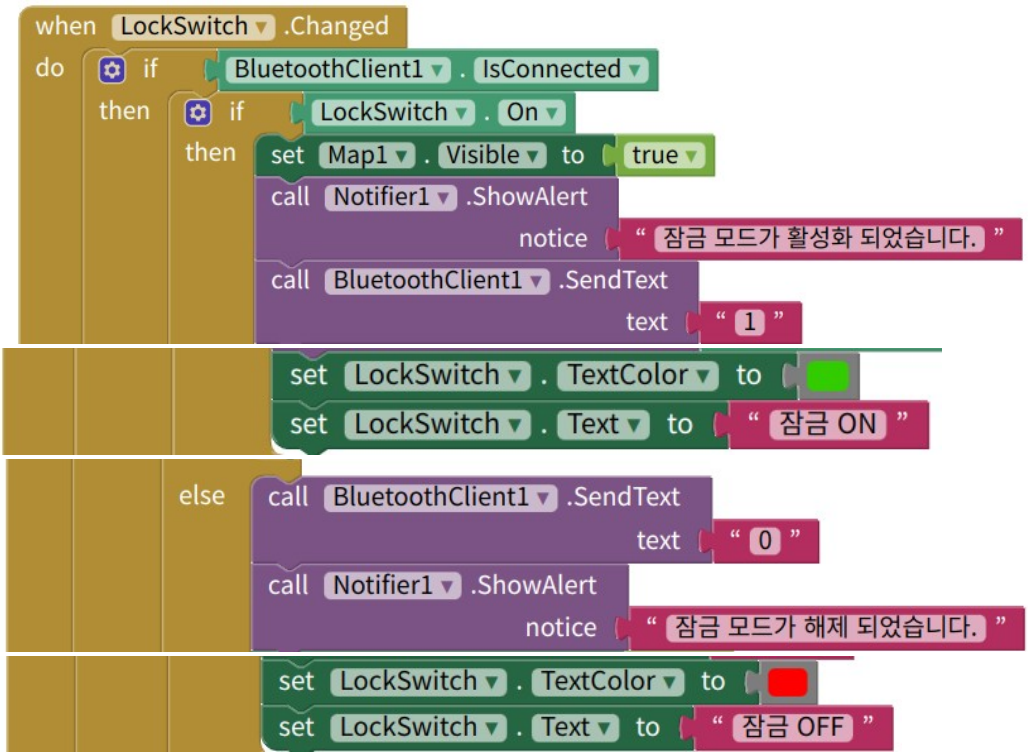
thread id 가 1 인 피에조 thread 인 경우 lock 이 해제되면 thread 도 종료하고 정상 상태를 Android 로 전송한다. lock 이 걸린 상태라면 피에조 부저소리를 계속 낸다.

(2) Android 시스템 구현

(2) - 1. BluetoothClient

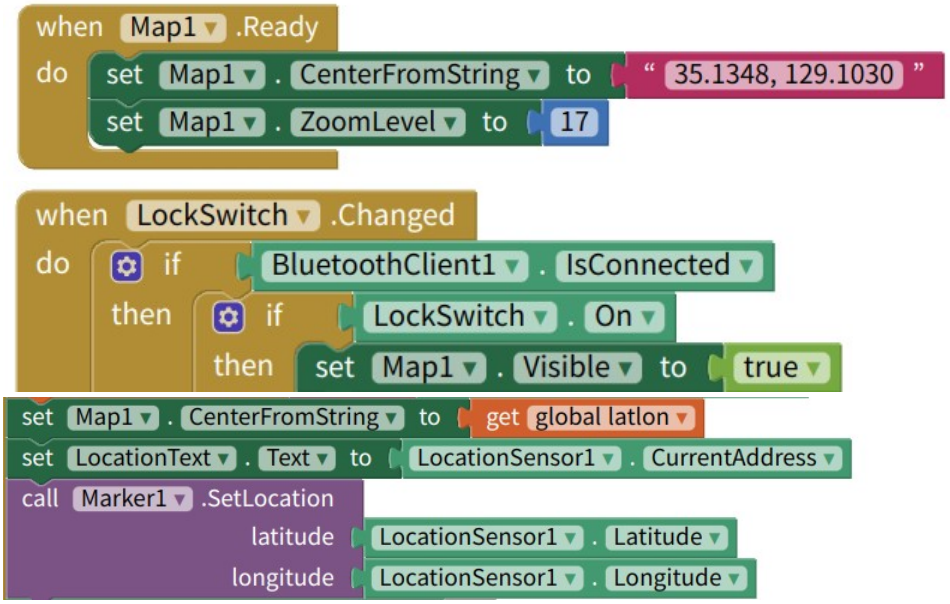


BluetoothPicker 를 이용해 Arduino Bluetooth 모듈을 BluetoothClient1 과 연결한다.



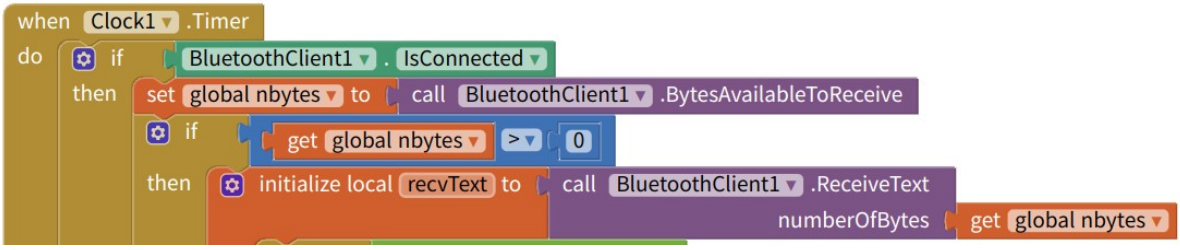
setCurrent() 함수의 역할을한다. BluetoothClient 를 통해 Arduino 로 잠금상태 1 또는 잠금 해제 상태 0 을 전송한다.

(2) - 2. Map



Map 의 초기위치와 zoomLevel 을 설정하고, LockSwitch 가 On 이 될때, Map 의 위치를 사용자 휴대폰의 현재위치를 설정하고 Marker 를 설정한다.

(2) - 3. Clock



App Inventor 의 Clock 은 Thread 와 역할이 비슷하다. BluetoothClient 가 연결되었다면 BluetothClient 가 Arduino 로부터 전달받은 값을 계속해서 읽는다. 이를 지역변수 recvText 에 저장한후 Notification Block 에서 활용한다.

(2) - 4. Notification

아두이노로부터 받은 recvText 가 1 이라면(측정 값이 기준치를 넘어 이상 변화가 생겼다면), 휴대폰 화면의 현재 상태를 정상 상태에서 위험 상태로 변경하고, 체크 표시 이미지를 X 표시 이미지로 변경한다.
이후 확장을 적용하여 생성한 Itoo1.Notification 을 통하여 백그라운드 알림을 받을 수 있도록 구현하고, 경고 알림음을 통해 사용자가 인지할 수 있도록 하였다.

---

4. 시스템 한계

(1) 아두이노 메가의 크기

아두이노 메가의 경우, 자물쇠 A 에 비하여 매우 가볍고, 비교적 높은 휴대성을 가지지만, 자물쇠 B 를 단독으로 사용하는 것과 비교하였을 때에는 더 무겁고, 휴대성이 떨어진다는 한계점이 존재한다.
하지만 이러한 한계점은 만약 실제로 제품을 설계하게 된다면 아두이노 메가 대신 아두이노 우노, 아두이노 나노와 같은 비교적 작고 경량화된 제품을 사용하여 극복할 수 있다고 생각한다.

(2) GPS 모듈 사용 불가

프로젝트 구현 과정에서 사용할 센서를 수령한 후 각각을 테스트 하였을 때, 문제가 없는것을 확인하였으나, GPS 모듈(NEO-6M)과 가속도 센서(MPU-6050)을 함께 사용할 시 값을 받아올 수 없는 현상이 발생하였다.
GPS 모듈과 블루투스 모듈만을 사용하였을 때 위도, 경도 값을 정확하게 받아올 수 있었고, 가속도 센서와 블루투스 모듈만을 사용하였을 때에도 가속도 센서 측정값을 성공적으로 받아올 수 있었다.

하지만 GPS 모듈과 가속도 센서, 블루투스 모듈을 함께 사용하였을 때, 작동이 멈추고 값 전달이 중단되는 현상이 발생했다.
우리는 이러한 문제를 해결하기 위하여 코드 변경, 아두이노 우노를 이용한 소프트웨어 시리얼을 이용, 가속도 센서 라이브러리 변경, RX/TX 를 업로드 이후 연결 등 생각나는 모든 방법을 시도 해 보았으나 문제를 해결하지 못하여 부가적인 기능을 담당하던 GPS 모듈을 설계에서 제외하기로 하였다.

(3) 블루투스 모듈의 거리 제한

이번 프로젝트 설계의 주 목적은 자전거 라이딩을 나갔을 때, 잠깐 가까운 거리에 대하여 자리를 비워야 하는 경우, 절도 범죄로부터 안심하고 다녀올 수 있도록 하는 것이다.
블루투스 거리 제한으로 인하여 연결이 끊어져도 피에조 부저를 이용한 경보음을 작동시키는 방법을 활용하여 보안성을 유지할 수 있도록 하였으나, 경보 작동 시 스마트폰으로 알림을 받을 수 없다는 한계점이 존재한다. 해당 문제는 실제 구현 시 HC-06 보다 높은 버전의 모듈을 사용하여 최대 통신 가능 거리를 늘리거나, 와이파이 모듈을 활용하여 해결할 수 있을 것으로 예상된다.

---

5. 최종 결과물 및 시연 동영상

(1) 가속도 센서 탐지 시연 동영상

https://github.com/gbdngb12/Country-Of-Eom-Bok-dong/blob/main/Video/Gyro.MOV

(2) 자석 센서 탐지 시연 동영상

https://github.com/gbdngb12/Country-Of-Eom-Bok-dong/blob/main/Video/Magnet.MOV

---

6. 참고 자료

<자전거 절도범죄 관련 기사>
https://www.domin.co.kr/news/articleView.html?idxno=1348197
<앱 인벤터 백그라운드 푸시 알림 Extension>
https://community.appinventor.mit.edu/t/free-background-tasks-itoo/55125
<피에조 부저 사이렌 소리>
https://create.arduino.cc/projecthub/sarful/arduino-workshop-piezo-sounder-alarm-5056d3
<MPU-6050 사용 예제>
https://blog.naver.com/PostView.naver?blogId=eduino&logNo=221081288204
<아두이노 스레드 사용>
https://github.com/jlamothe/mthread
<아두이노 보드 별 I2C 핀>
https://blog.naver.com/roboholic84/220583168600
<GPS 모듈 사용 예제>
https://m.blog.naver.com/cherrychance/221809429310

---

7. 소스 코드

https://github.com/gbdngb12/Country-Of-Eom-Bok-dong